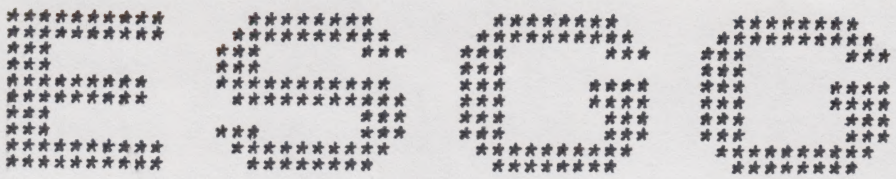
Bimonthly periodical of the "Exidy Sorcerer Gebruikersgroep"



The LOGICAL partner for a Sorcerer

Subscriptions: annual f. 18.00 Holland & Belgium
f. 22.50 Europe
f. 27.00 Other countries

Officers
Pres.: F. Vogelaar, Karnemelksloot 197, NL-2806 BE Gouda, Holland
V.Pres.: W. Jonker, A. Franklaan 681, NL-1443 HV Purmerend, Holland
Secr.: J. Netteler, Pr. Hendrikstr. 3d, NL-3071 LG Rotterdam, Holland
Treas.: C. Peeters, Rubicondreef 100, NL-3561 JE Utrecht, Holland

Coordinators
Softw.: W. Warning, Dotterbloem 32, NL-8265 HG Kampen, Holland
Hardw.: A.v.Duyvenbode, Eliotplaats 7, NL-3068 EE Rotterdam, Holland
P.R. Intern.: C. Boone, Stationsplein 26, B-9100 Lokeren, Belgium
Literature: S. Oegema, Florisdonk 10, NL-4707 VM Roosendaal, Holland
Obj.Groups: F. Cieremans, Veldzichtln 29, NL-3461 GT Linschoten, Holland
Editor: ESGG, Postbus 510, NL-1000 AM Amsterdam, Holland

* *

* * * * * C O N T E N T S * * * * *

Copyright 2
Advertisements 2
ESGG Service 2
Debug Modification 3
Input 4
ESGG Mini Inquiry 4
Number Lay-out..... 4
From the top 4
Data Poke Program 5
Basic Extension 11
Rectifications 15
SUBSCRIPTION FORM 15

* *

COPYRIGHT

Copying articles of this issue is only permitted if you mention where they come from: E.S.G.G. VOL. 1 N° 2 PAGE ... and only when you are a non-commercial subscriber.
Others must first get a written permission from the editor (address on page 1)

* * *

ADVERTISEMENTS

Non-commercial ads are welcome on following conditions::

Format: 1 line = 66 characters or spaces
Max. 6 lines per ad
Price: 3 Dutch guilders for 2 lines
6 Dutch guilders for 4 lines
9 Dutch guilders for 6 lines
Address: send your text to the editor, together with your payment to postal cheque account 2058443 of HCC-ESGG, Utrecht, Holland

* * *

Would you like to receive this paper regularly?
Then please use the subscription form on the last page.

* * *

E.S.G.G. SERVICE

Our Sorcerer Day in Gouda (Holland) was such a success, that many Sorcerers couldn't get what they had come for.

We now have enough for everybody and all Sorcerers are welcome on 4 September in Rotterdam, at the Tom Menken Hal, near the Central Station.
It's also possible to get what you want by post.
Order by paying on our postal cheque account 2058443 for HCC-ESGG, Utrecht, Holland. Please clearly mention what you want!
What can you get?

ARTICLES (prices in Dutch guilders for 1 piece!)

	If you come and get it	By post
Cassettes with programs(1-5)	7.50	10.00
Basic Extension (EPROM)	20.00	25.00
Inverse video print (kit)	7.50	12.50
Single issues of this paper	3.50	4.50

* * *

DEBUG MODIFICATION

Sorcerer De Witte from Heiloo (Holland) sent us a small modification for the DEBUG program. Let's try!
After loading, quitting "warm" by hitting 'L' (RETURN)
Then enter CD Ø3 EØ (RETURN)
G (RETURN)
Then hit 'S' and after that CTRL Q or SHIFT H, after which the cursor is "home". Then you can quit via CLEAR or stay in DEBUG by hitting the space bar.

* * *

INPUT ... for asking questions

Send your questions about your hobby to the editor.
We don't know everything, but we'll do our best to answer your questions.

* * *

ESGG MINI INQUIRY

There were two questions:
1. Are you interested in an insurance for your system?
2. Would you like an own paper?

The answers:
1. 74.3 % said NO.
2. 98 % said YES.

Conclusions:
1. No insurance.
2. Our own paper.

* * *

NUMBER LAY-OUT (again)

This little program comes from Sorcerer Kuijper

```
10 INPUT A:A$=STR$(INT(A))
20 X=LEN(A$)
30 PRINT TAB(26-X):A:GOTO 10
```

* * *

FROM THE TOP

Our members know the names and addresses of our officers and coordinators. Yet they don't know enough about each of them, with the exception of the software manager, who has already brought out 5 cassettes, full of software.

We'll try to give some more information about each coordinator. Today, we present the coordinator for the "object groups". Members who are working in some special field should inform the coordinator, so that he can inform other members who are interested in the same field.
This can lead to "object groups" and the idea is not to invent the wheel again and again.

* * *

DATA POKE PROGRAM: DAPO5

Rob de Beer from Capelle aan de Yssel (Holland) sent us a program, intended to provide data to a Basic program, working with a machine language program.
These data are already in the machine language program and are fetched from it to be put in a data area in the Basic program. Let's listen to Rob.
The data list is present in decimal form in the machine language program. Each hexadecimal byte is replaced by a number between Ø and 255. By means of a simple FOR-NEXT loop with READ and POKE, Basic can put the machine language program anywhere in memory.(See the last 3 statements in line 145)
With the Basic USR-function, the machine language program can be called at will. The charm of this technique is that only ONE Basic program must be loaded to get everything working. The program index also becomes simpler. It's a good old method, but still it has some disadvantages.
The memory for the machine language program is thrice as large as when you load it separately, but this will not be a problem very often. Converting long machine language programs to decimal is a more serious problem, as typing the data list is tedious and takes a lot of time. Even when working with full concentration, errors are likely to occur.

I hope that DAPO5 will prove useful, because the hexadecimal bytes can be typed in, after that DAPO5 itself fills in the data list, puts the machine language program where you want it in memory and gives a hexadecimal dump for checking, plus a listing of the data list, with the start address in the first line. DAPO5 has not been written for a specific application. I was fascinated by the idea that a Basic program could change or implement itself. Then I tried to find an application where this could be realised. I'd like to see other articles on this subject from people who are experimenting with it. Variations are possible or even a completely different method to combine a Basic program with a machine language program. In March 1981 Eeltje de Vries wrote a magnificent article about it in the Hobby Computer Club Newsletter.
A variation could be a program with a hexadecimal data list, so that it becomes interesting for advanced machine language programmers. Or a program which fills the data list from a machine language program that already resides in memory.
The heart of the "self changing" program consists of lines 44,130-145,180 and 184. Further the six data lines must be prepared, but less lines will do. Each of these lines has 2 spaces, followed by 54 commas.
With POKE-statements you can change a Basic line, but you cannot make it shorter or longer; you can join two lines or split a line or seem to shorten it. Much is possible, as long as the POKEs get on the right spots. Their effectiveness must be maintained in this program, also after the Basic program would have been changed "normally"!
The POKEs which the Basic program executes I would like to call PIPs (program implementing poke). The PIPs may appear anywhere in the program, e.g. in line 164 of DAPO5:
POKE 479,CL: here address 479(ØlDF Hex) gets a content equal to the value of CL. ØlDF is 10 locations further than the Basic program start address (ØlD5) and hence must be line Ø, which starts with REM, followed by 11 bytes of

machine language: ØC 11 vv ww 21 yy zz CD F8 E4 C9.
The last ten bytes load the DE and HL registers and the jump to address E4F8 in the monitor. Here is the monitor routine DUMP, which we need for the hexadecimal dump from address (DE) to address (HL).
It's important that the PIP to address 1DF puts the value of CL in the place of what is yy above.
In other words, the 11 machine language bytes in line Ø should never be moved.
I don't think this would harm the flexibility of the Basic program, because sensible changes will leave line Ø as it is. Also because of the USR call, line Ø must not be changed!
Now the effectiveness of the PIP in line 164 seems all right (also thanks to the neighbourhood of address Ø1D5). The situation quite changes if we take Ø1D5 as a starting point for a change in line 2C..Each REM which would be added or deleted between lines Ø and 2Ø would make the PIP land on the wrong place! So we must use a safer starting point, which we will call RA (reference address). You can always find such an RA. We'll illustrate this with the PIPs which fill in the data list of DAPO5.
In line 7O the hexadecimal byte H$ with number D changes into the decimal number C, after the conversion of hexadecimal into decimal. In line 94 we make of this a string G$(D), which must go to the data list and may have a length E of 1,2 or 3. That happens in lines 13O,135,14O and 145, with each step of I. B starts with the value Ø. With each step of J, a character of G$ is being POKEd into the data list in address Z+S. Mostly the character there takes the place of a comma.
S is a counter, which increments at each step of I and J as well, so that after each string G$ a comma remains in the data list.
Z is what has been called RA and is obtained with line 44. Remember that Basic keeps a register on locations 445-446, which points to the addresses of the last reading of the data list.

So after 44 READ R$ this would be the place of the extreme right y in data line 200? No! One spot further, because the READ instruction only stops after reading the comma of ASCII Ø which is found at the end of each Basic line.
And now let's count: 201 REM takes six memory locations and 202 DATA takes 5, together 11. In line 44 Z is already incremented with 2 and in line 91 with 11, together 13.
The first character we POKE (or rather PIP) with line 145 comes 2 spots after DATA in line 202, right in front of the first comma. Etc.,etc.
Line 201 REM is only there to make the data list more visible and in fact is against the rule that the PIP must be as close t the RA as possible.
If you would delete line 201, all PIPs would come six spots further and then the program will make errors, because of the first five commas. For safety, line 201 should become a data line. Please also change line 91 Z+11 into Z+5!
Then line 135 which prevents that at the end of a data line the numbers (in fact strings) should be cut off. So there remain Ø, 1, 2 or 3 commas, which are replaced by colons(:) by the PIP in line 180. As far as I can see the ':' is the only character that works without errors, when reading as well numerical as string constants.
At last, line 190 contains 2 PIPs for line 200. It's evident that for this a RESTORE and READ R$ was necessary. This is an example of a text modification by using PIPs. There is more to be told, e.g. the last statement in line 130, which makes that Sorcerer's negative POKE limit can be transgressed correctly at address 8ØØØ H.
I hope that the essence of the program is now clear. I'd like to add this. Imagine that you've filled in your data list with DAPO5 and you want to keep this list, but separate of all what precedes. In stead of deleting line after line, you could do this:

PRINT PEEK(445);PEEK(446). I then see 75 11. Subtract 61 and there remains 14 11, which is address BØE. Then BYE.
< EN 1D5 < E (CR) B (CR)/ (CR) and the 1D5-1D6 linkpointer is on BØE, the start address of the data list.
Then go on with PP,LIST and Ø (CR) and ready you are!

And this is the program DAPO5

```
000 REM !!!!!'C'I5F4&
001 REM ATTENTION PLEASE! In line Ø are 10 bytes of machine
002 REM language, which are necessary for the hexadecimal
003 REM dump. DON'T CHANGE ANYTHING!
020 REM DAPO5 is the name of this program, 81 12 03
021 REM Modified 81 12 17
022 REM Application: BASIC + machine language program
023 REM Give RAM start address and type in the hexadecimal
024 REM machine language bytes. End with /.
025 REM Then DAPO5 will do the rest, i.e.
026 REM Ø.Blocks input if not hex. byte, line 76.
027 REM 1.Hex/dec. conversion, lines 73,82,85,94.
028 REM 2.Filling the data list with converted bytes.
029 REM 3.Puts mach.lang. at given RAM address, lines 88,145.
030 REM 4.Hex.dump for final checking, lines Ø,88,91,160,164.
031 REM 5.DATA line 200 text modificn, lines 79,190.
032 REM 6.Listing of finished DATA list, lines 48,190.
033 REM 7.Line Ø protection, line 170.
034 REM
035 REM Each DATA line has 2 spaces, followed by 54 commas.
038 REM * Author: Rob de Beer, Capelle aan de Yssel, Holland *
039 REM * Telephone 010 - 505435 *
040 CLEAR 999:DIM G$(200):I$="Now type in hexadecimally:"
044 READ R$:L=PEEK(445):H=PEEK(446):Z=256*H+L+2:REM READ address!
048 IF F THEN R$="And this is now the DATA list:"
052 PRINT:PRINT TAB(X)R$:J$="xx ":V=-4:IF F THEN RETURN
```

```
056 PRINT:PRINT I$;J$;:INPUT H$:GOSUB 73:J$="yy ":IF V THEN 56
060 PRINT:M=23:X=13:K=G:F=G-65536:GOTO 68
064 PRINT,"WRONG! Please type in again":ON 1-V GOTO 70:RUN
068 PRINT"Type in the HEXADECIMAL bytes. Stop with /":PRINT
070 PRINT D;:INPUT"HEX ";H$:REM Start of hex/dec conversion
073 C=Ø:W=LEN(H$):A=ASC(H$):IF A=47 THEN K=K+D:PRINT:GOTO 130
076 IF A>57 AND A<65 OR A<47 OR A>70 OR W=Ø OR W>2 THEN 64
079 IF V THEN POKE Z-W+V,A:REM Filling in start address
080 REM of DATA line 200
082 C=C+VAL(LEFT$(H$,1)):IF A>64 THEN C=C+A-55:REM ASCII 65=A
085 IF W=2 THEN C=C*16:H$=MID$(H$,2):A=ASC(H$):W=1:GOTO 76
088 IF V THEN G=G+C:V=V+2:IF V THEN G=G*256:POKE 477,C:RETURN
091 IF F=Ø THEN POKE 476,C:Z=Z+11:RETURN:REM Loading the DE register
094 G$(D)=MID$(STR$(C),2):PRINT,CHR$(23)G$(D):D=D+1:GOTO 70
130 FOR I=B TO D-1:G$=G$(I):E=LEN(G$):IF F++2*I=>Ø THEN G=F
135 IF E>53-S THEN GOSUB 180:N=Ø:S=Ø:B=I:Z=Z+62:GOTO 130
140 FOR J=1 TO E:A$=MID$(G$,J,1):S=I-B+N
145 POKE Z+S,ASC(A$):N=N+1:NEXT J:READ R:POKE G+I,R:NEXT I
160 PRINT TAB(X)"Here comes the hexdump":C=INT(K/256)
161 CL=K-256*C
164 PRINT:POKE 479,CL:POKE 480,C:POKE 260,218:POKE 261,1:U=USR(Ø)
170 FOR J=476 TO 480:POKE J,33:NEXT:REM No zeroes in line Ø!
180 T=T+1:IF PEEK(Z+S+T)=44 THEN POKE Z+S+T,58:GOTO 180
184 T=Ø:IF I=D THEN RETURN:REM Deletes the remaining commas.
190 RESTORE:GOSUB 44:POKE Z-31,101:POKE Z-28,30:LIST 200
199 REM
200 DATA This machine language program is set at start address xxyy
201 REM
202 DATA ,,,
```

BASIC EXTENSION TAPE ROUTINES

The aim of these routines is to CLOAD* and CSAVE* without any problem and to be able to load and save alphanumerical data from and to cassettes.
These routines use an auxiliary buffer which starts at address FCØØ and should not exceed 1280 bytes. The buffer then goes from FCØØ over FFFF to ØØFF.
You may have noticed that this is the Sorcerer's graphics area and page Ø. If 512 bytes will do, you can still use the graphics. Then you should, after having used the routines, use CHR$(12) to restore the standard graphics, while the other graphics remain unchanged. The normal Basic memory area remains intact, even if you need 1280 bytes.
Now we can save name, address and city together to tape, so that loading and working on part of the database is possible.
You are not allowed to work with empty strings. You should check this in your application program (see line 1120, from IF TP$=" " to the end and line 1152.)
If anyone can still squeeze BEXT together, we'll gladly accept suggestions.
For saving you can choose between recorders 1 and 2 (see lines 1340-1345). Loading is only possible from recorder 1.
ATTENTION! The variable you use for your tape routines should absolutely be named first in your Basic program. (See line 10). In the example we use TP$, but of course any other $ variable will do.

Example program

Lines 10-110 Opening the database and choosing between loading and saving
Line 101 changes TP$ into capitals
Lines 745 and 748 application of the "GET" routine

Lines 1110-1125 Saving routine
Line 1115 Check if buffer is full
Line 1116 Full buffer is being saved
Line 1118 Closing the database
Lines 1120-1125 1 STRING to the buffer
Lines 1130-1155 Loading routine
Lines 1340-1350 Choice of recorder + screen message
Lines 1440-1445 Defining the memory needed for 1 element

The example is part of a database program with the following variables:

B: start of database E: end of database
L: number of fields per record IT: number of filled records

In this example the buffer takes 1280 bytes (see line 1116)
Auxiliary values are: CHR$(144) indicates end of database
CHR$(147) is a dummy for an empty string
In Basic it means STOP and NULL.

```
0010 CLEAR 9999:TP$=" "
0090 INPUT "Name of the database (5characters)";TP$:T=USR(5)
0100 PRINT "Loading (default) or saving?":GOSUB 745
0101 T=USR(2)
0105 IF TP$="S" THEN GOSUB 1340:GOTO 1110
0110 C=1:TP$="1":GOSUB 1345:GOTO 1130
0745 T=USR(7):T=PEEK(-4041):IF T=Ø THEN 745
0748 TP$=CHR$(T):RETURN
1110 FOR I=B TO E:GOSUB 1440
1115 C=PEEK(-4043)*256+PEEK(-4044)+LT
1116 IF C>1279 THEN T=USR(6):GOTO 1115
1118 GOSUB 1120:NEXT I:TP$=CHR$(144):T=USR(3)+USR(6):RETURN
1120 FOR K=1 TO L:TP$=X$(I,K):IF TP$=" " THEN TP$=CHR$(147)
1125 T=USR(3):NEXT K:RETURN
```

```
1130 I=IT+1
1135 GOSUB 1150:IF TP$=CHR$(144) THEN IT=IT-1:RETURN
1140 I=I+1:GOTO 1135
1150 FOR K=1 TO L:T=USR(4):IF TP$=CHR$(144) THEN RETURN
1152 IF TP$=CHR$(147)THEN TP$=" "
1155 X$(I,K)=TP$:NEXT K:RETURN
1340 PRINT "Which recorder?(default=1):GOSUB 745:C=Ø
1342 IF TP$<>"2" THEN TP$="1"
1345 POKE-4064,ASC(TP$):PRINT CHR$(12);"Get recorder";"TP$ ready for";
1348 IF C THEN PRINT "playing";:GOTO 1350
1349 PRINT "recording";
1350 PRINT" and hit any key.":GOTO 745
1440 LT=Ø:FOR K=1 TO L:LI=LEN(X$(I,K)):IF LI=Ø THEN LI=1
1445 LT=LT+LI+1:NEXT K:RETURN
```

The USR routines 2,3,5 and 6 act on the momentary value of the first named variable (in the example TP$) USR(4) gives that variable the value of the next string in the (read) buffer.

Now something about starting up. Exidy's Standard Basic remains unchanged in your PAC. You can start the Extension by using the Basic warm start. Hold the ESC-key down. Then hit both RESET-keys shortly and at the same time. Let the ESC-key come up, when you see "READY" on the screen. The Extension is now started.
You can also use another method: BYE (CR), PP (CR).
When starting the Extension, 2 things happen:

1. The values on the USR vector are put to the start address of the USR selection routine (Ø1Ø4/Ø1Ø5 hex.;260/261 dec.)
2. The input vector is put at DF37 hex. (SE I=DF37).

So all Basic input goes through the correction routine. This may conflict with some existing programs, which need an own USR vector.

In those cases you should POKE the correct values at the beginning of such a program: POKE 260,... and POKE 261,...
E.g. for KISS2: line 5 POKE 260,51:POKE 261,188.
The last value (188) is for 48 K. For 32 K. it should be 124 and for 16 K. 60. The new USR routines cannot be used at the same time. CTRL-keys can.
A program may need a specific INPUT driver. This kind of programs often enter Basic via the vector for a warm start and so start the Extension, eliminating their own needed driver. To avoid this, these programs should jump directly to Basic warm, and not via the vector. E.g. in Toolkit, you change the addresses X89E and X89F of FA and DF (vector DFFA) to 6B and CØ (Basic warm CØ6B). X is for 48 K. B, for 32 K. 7 and for 16 K. 3.

We'd appreciate if you could let us know your modifications, so we can inform all our members. Sorcerers help Sorcerers!

Another, more rigorous method is to abandon the possibility to start the Extension by a warm start. In that case the contents of the EPROM addresses DFFB and DFFC, where now you find EØ and DF (hex. DFEØ), should be changed into 6B and CØ (hex. CØ6B). BEXT can then be started from monitor (BYE): GO DFEØ. If you don't have an EPROM programmer try to contact another Sorcerer who has.
If you want to trace a whole Basic program, you must start TRACE with T=USR(10):RUN. If you want to trace parts of a program put T=USR(10) just in front of the wanted part and at the end you put T=USR(11), which stops TRACE AND BEXT!
So after TRACE you have to start BEXT again.
Before each INPUT statement TRACE has to be stopped. E.g.
100 X=USR(11):INPUT A$:X=USR(10)
So: put out, input, put on again.

RECTIFICATIONS

In ESGG 1 is said that the USR functions are always ready. It is true, if you first set the USR vector. This happens automatically when starting BEXT, but can also be achieved by poking value 88 in address 260 and 223 in address 261. The correction routine can also be activated separately by the monitor command SE I=DF37.
In the description of CTRL B FØ21 should be deleted. The value in that address is being used as "delimiter".
In line 2110 of the program example from ESGG 1, the PEEK values are incorrect. In this issue's example the correct values are printed in line 1115 (-4043 and -4044).

That's enough for now. Next time we'll answer your questions and we'll publish the memory map.

Wim de Kreuk and Floor Vogelaar

.....

OK, I want to subscribe to "ESGG". I'm paying 22.50 Dutch guilders (inside Europe) or 27 Dutch guilders (outside Europe) to your postal chegue account 2058443 for HCC-ESGG in Utrecht, Holland.

My full name: .....
My complete address .....
ZIP Code ..... City .....
State ..... Country .....
Date ..... Signature .....
Don't write here: E 282 Datum ontvangst: . . / . . / . . . .